Protocol Boosters

A Kernel-Level Implementation

A Hardware-Level Implementation

And then some .....

W. S. Marcus

Senior Research Scientist

Telcordia Technologies

wsm@research.telcordia.com

My Objectives

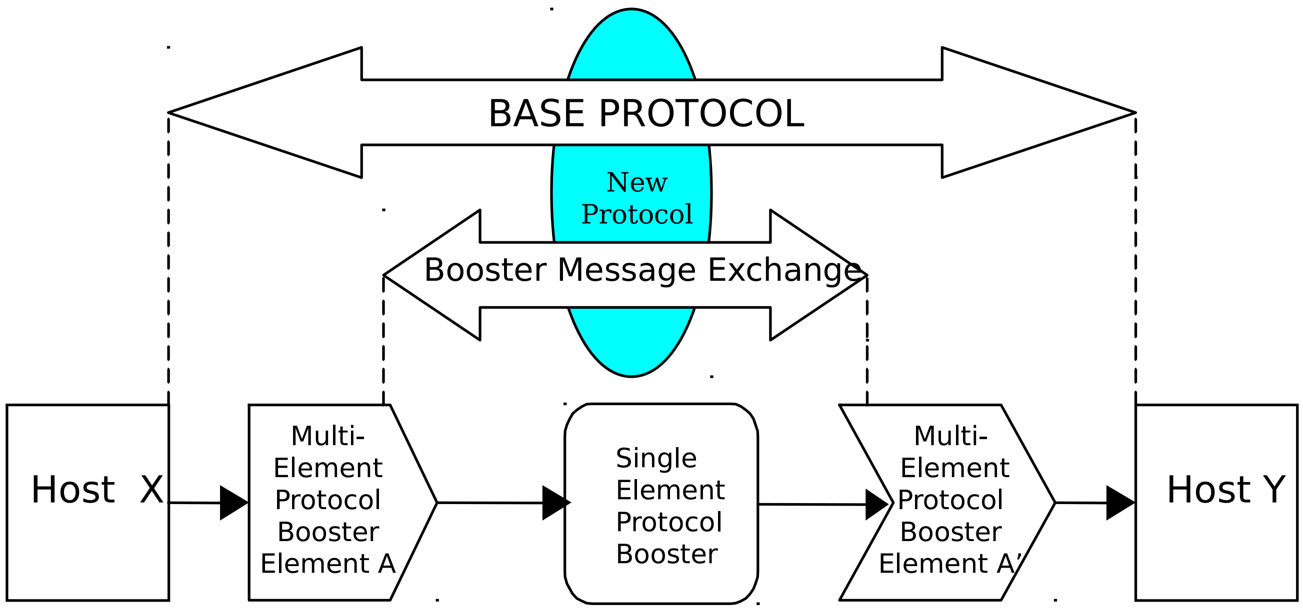
- Introduce Protocol Booster design methodology
- Describe a kernel-level implementation
- Describe a hardware-level implementation
- Highlight applications and results

Telcordia™ Technologies
Formerly Bellcore… Performance from Experience

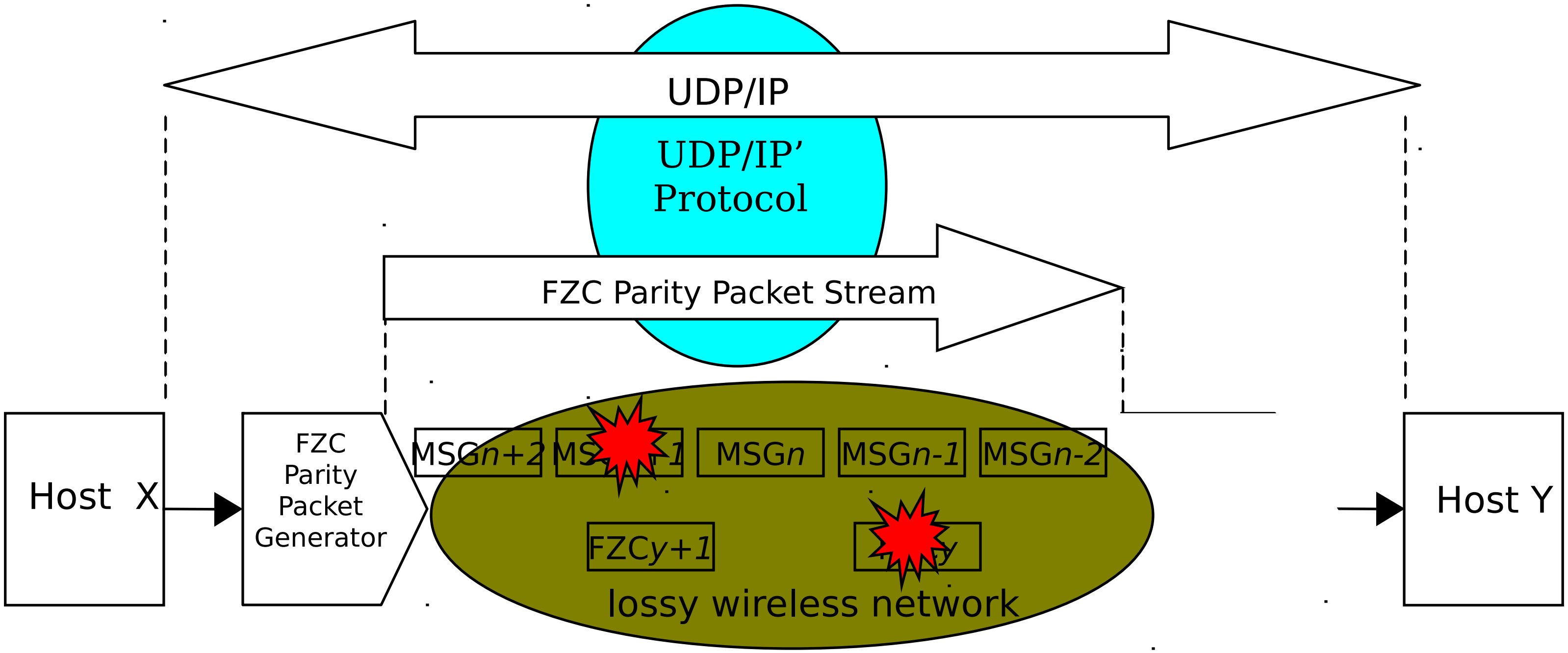
Protocol Boosters:
What are they?

- Conceived as software/hardware modules that:
  - *Transparently, selectively,* and *robustly* enhance existing communications protocols.
  - Are used to *incrementally* construct new communications protocols.
  - Allow protocol design to *track* improvements in networking technologies.
  - *Reduces* inefficiencies associated with designing for the worst-case.
  - Permit *on-the-fly* adaptation/customization of a protocol.

Protocol Boosters:
What are they? (continued)



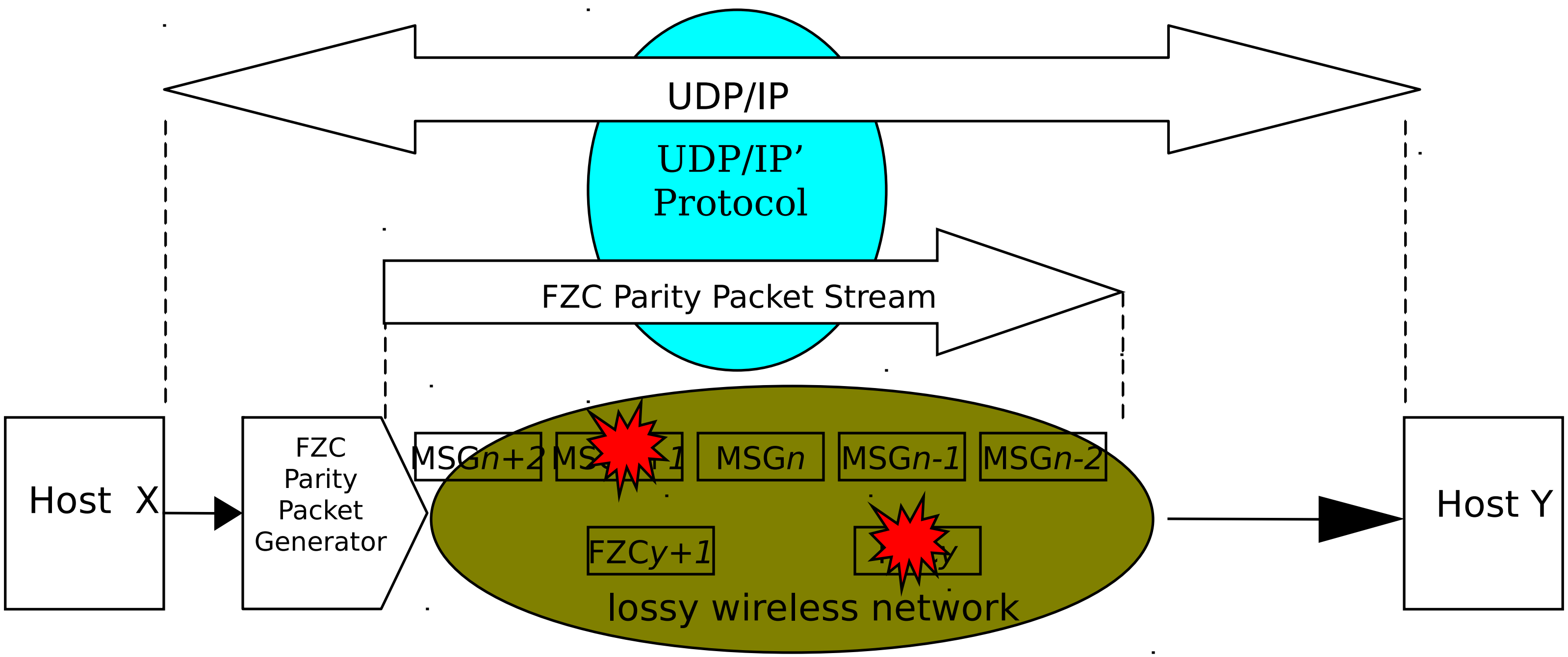
Protocol Boosters:
Example: FZC



Telcordia™ Technologies

Formerly Bellcore…
Performance from Experience

Protocol Boosters:
Example: FZC



Protocol Boosters:
Example: FZC (continued)

- Desirable for
  - Applications with latency constraints
  - Applications for multicast distribution
  - Applications that respond to packet loss with retransmissions
  - Networks with no, or slow, return channels
- Not desirable for congestion loss (use other boosters)
- Employs a novel FEC scheme that
  - Based on Reed-Solomon Code
  - Allows fast software implementation (Mb/s)
  - Receiver need not know the number of parities sent
  - Receiver need not wait for, nor calculate, any parity packets beyond the number of missing data packets.

Protocol Boosters:
Protocol Boosters and other adaptation technologies

- Link Layer Services
- Protocol Conversion/Termination
- ***Protocol Boosters***
- ***Active Networking***
- Basic differences
  - transparency
  - robustness
  - selectivity
  - dynamism
  - ease of deployment
  - flexibility
  - generality

Telcordia™ Technologies

Formerly Bellcore…
Performance from Experience

Protocol Boosters:
Some Guiding Principles:

- Protocol Boosters
  - Can reside *anywhere* in the network.
  - Can operate within the confines of a *single* network element.
  - Can be *distributed* over several network elements.
  - Can *add*, *delay*, or *delete* end-to-end messages.
  - *Never* modifies syntax or semantics of end-to-end protocol exchanges.
  - *Transparent* to protocol being boosted and elimination of any part of the booster does not, in itself, prevent end-to-end communications.
  - Are instantiated or revoked *on-the-fly* based on *policies* (e.g., observed network behavior, time of day, etc.)
  - Can be *nested* and *concatenated*.
  - Are an Active Networking programming model.

Active Networks:
General Principles:

- Architecture for supporting rapid, reconfigurable, and dynamic services *on a per packet basis!!!*
- Each packet can deliver new functionality into the interior of the network.
- Secure packet processing
- Safe packet processing
- Techniques for code mobility and service deployment

Telcordia™ Technologies
Formerly Bellcore… Performance from Experience

Protocol Boosters:
Booster Modules

- Monitoring
  - History, Trace
- Debugging
  - Add, Delay, Delete, Duplicate
- Error Control Family
  - ARQ-R, ARQ-A, ACK-COMP
  - FZC, ED, RO, DUP-DTEC, MSS
  - *FEC, DAT-COMP, SEC <-- Violate guiding principles*
  - *PMOD*

Protocol Boosters: *Implementations*

- *Adhere* to guiding principles.
- Focus on *performance*, e.g., below the EE/NodeOs line.
- *Kernel*-level implementation for boosting IP
  - Procedures followed by a protocol
  - ARQ, FZC, flow control, signaling etc.
  - Packet oriented processing
- *Hardware*-level implementation for boosting ATM at the OC-3c line rate.
  - Core functions used by a protocol
  - FEC, CRC checking and calculation, encryption, authentication, packet filtering, compression etc.
  - Bit oriented processing

Telcordia™ Technologies

Formerly Bellcore… Performance from Experience

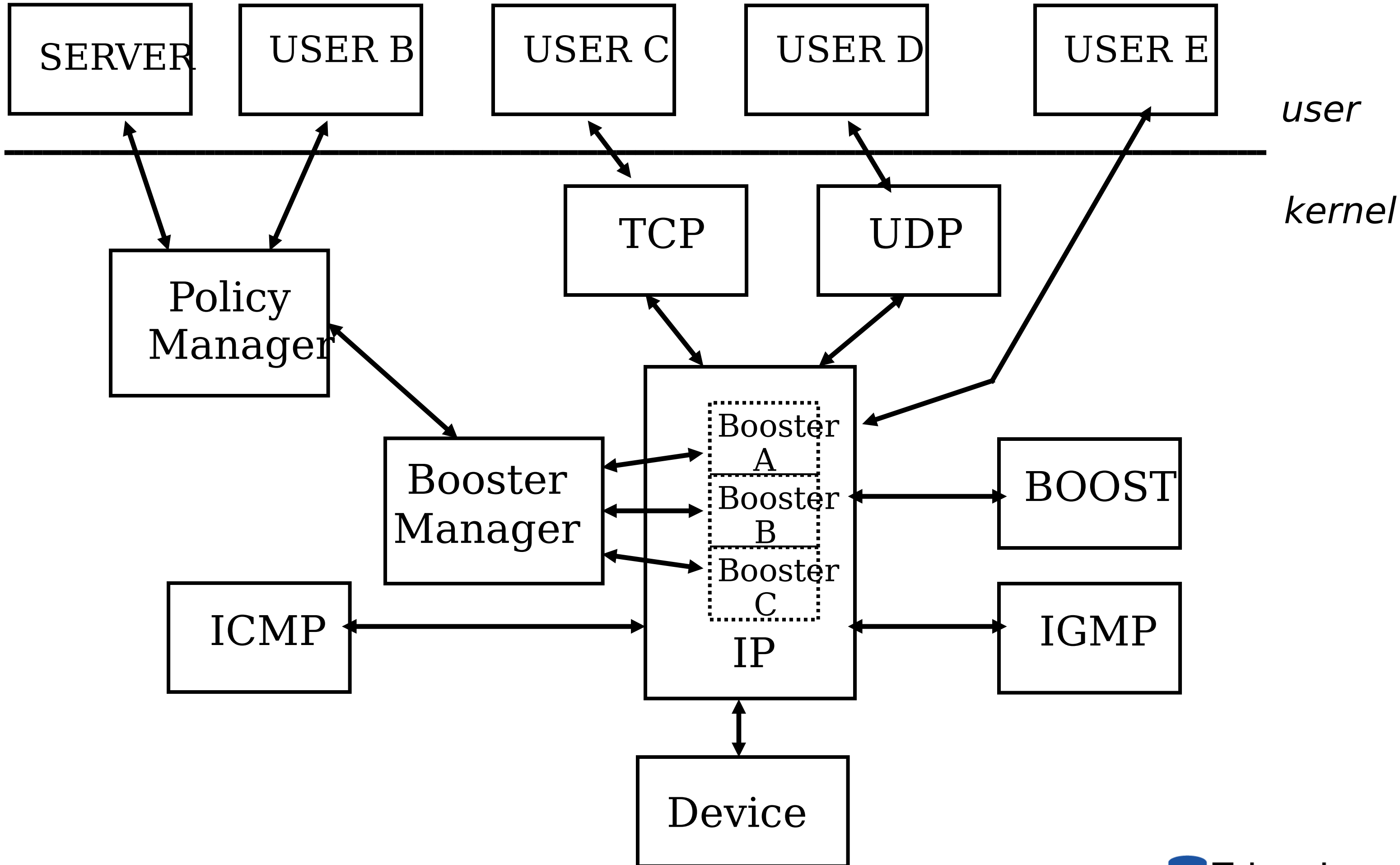
Protocol Boosters

Kernel-level implementation II: Linux 2.0.32

- Individual protocol booster modules are implemented as *loadable kernel modules, lkms*.
- Six interfaces per module
  - instance manager (includes /proc file system hooks)
  - input, output, and forward interfaces
  - booster channel interface (out-of-band channel)
  - ioctl interface
- A *policy manager* (lkm) uses a variant of the Linux IP firewall mechanism to construct flow traps directed at *sequences* of booster modules.
- A *booster manager* manages the loading/unloading of the individual boosters via the *kerneld* facilities, presents statistics via the */proc* filesystem, manages the flow of packets through the booster modules.
- Booster deployment via a client/server paradigm (presently under human intervention).

Protocol Boosters

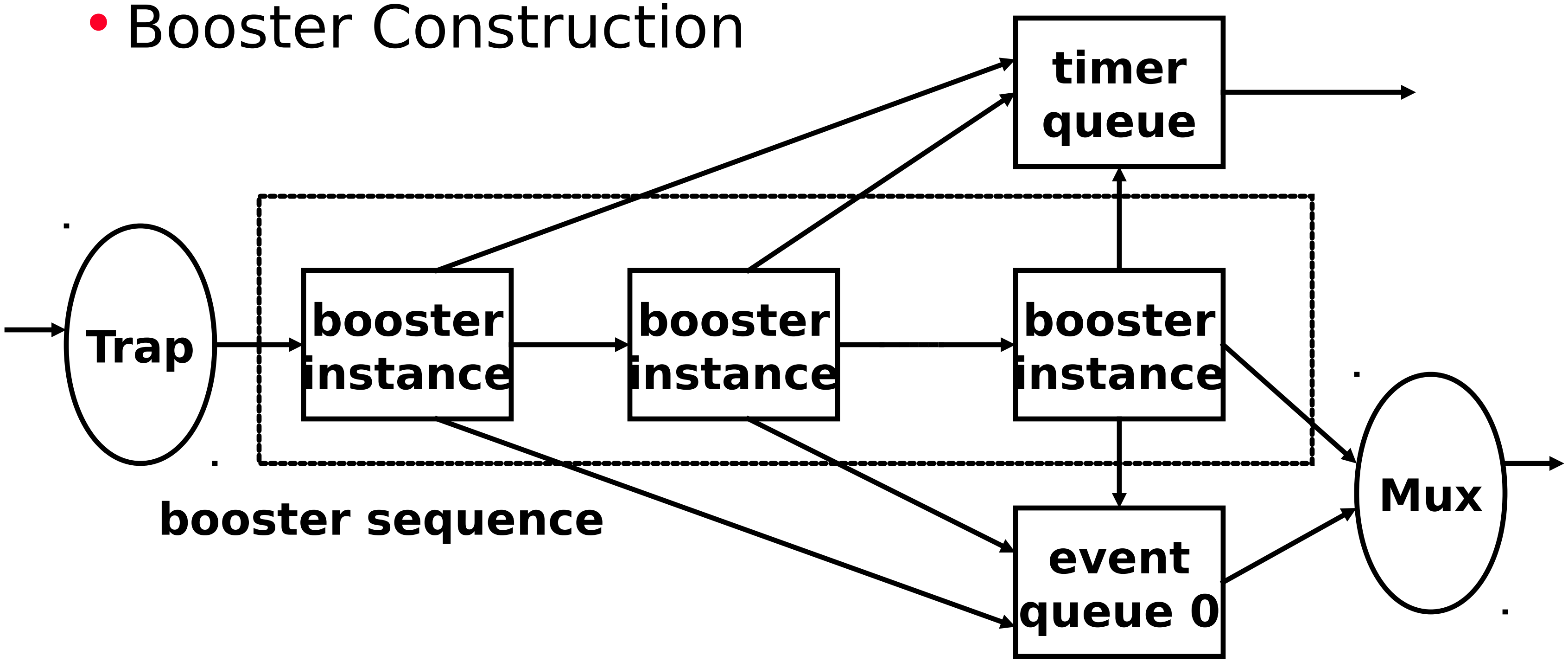
Kernel-level implementation II: Linux 2.0.32 (continued)



Protocol Boosters

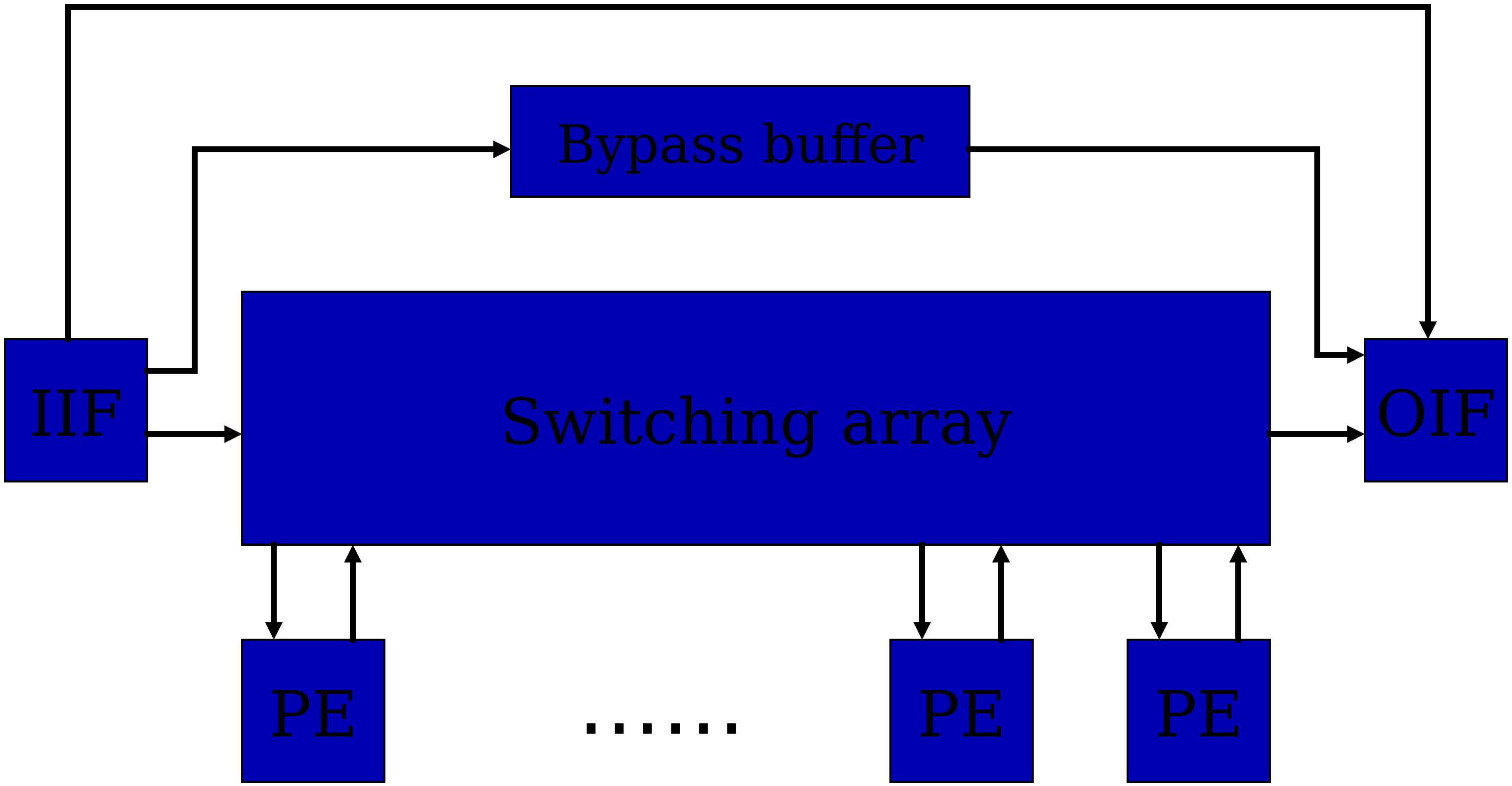
Kernel-level implementation II: Linux 2.0.32 (continued)

- Booster Construction

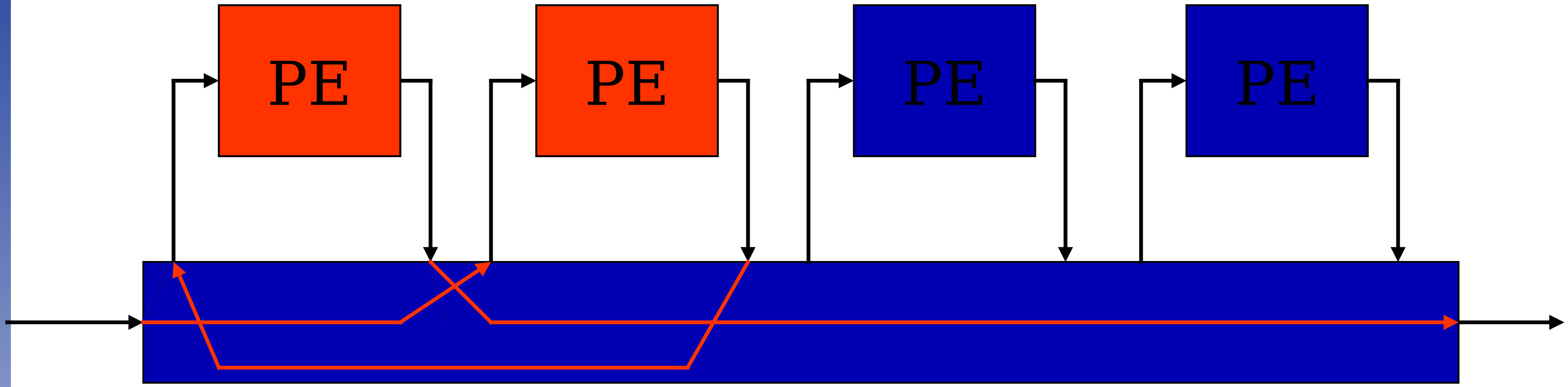


Protocol Boosters :

Programmable Protocol Processing Pipeline (P4)

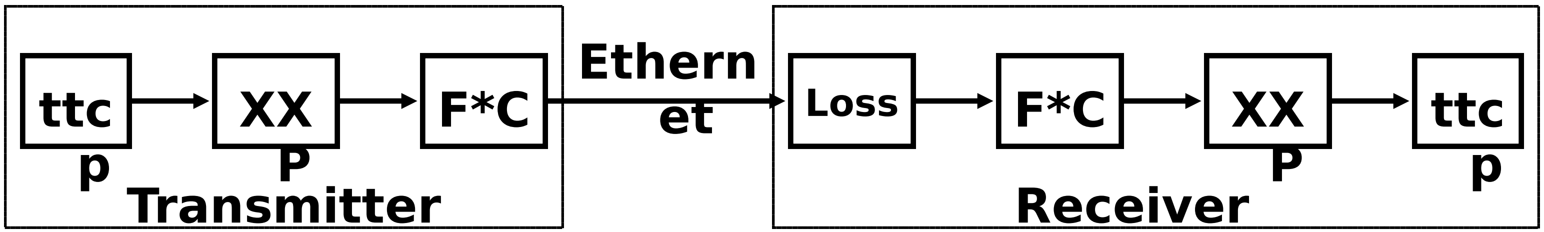


Protocol Boosters : *Programmable Protocol Processing Pipeline (P4)*



- SRAM (Altera 10K-series) based programmable devices
- Switching array (reconfiguration time = 1us)
- Processing implemented in hardware
- Dynamic reconfiguration during run time (device download time = 100ms)

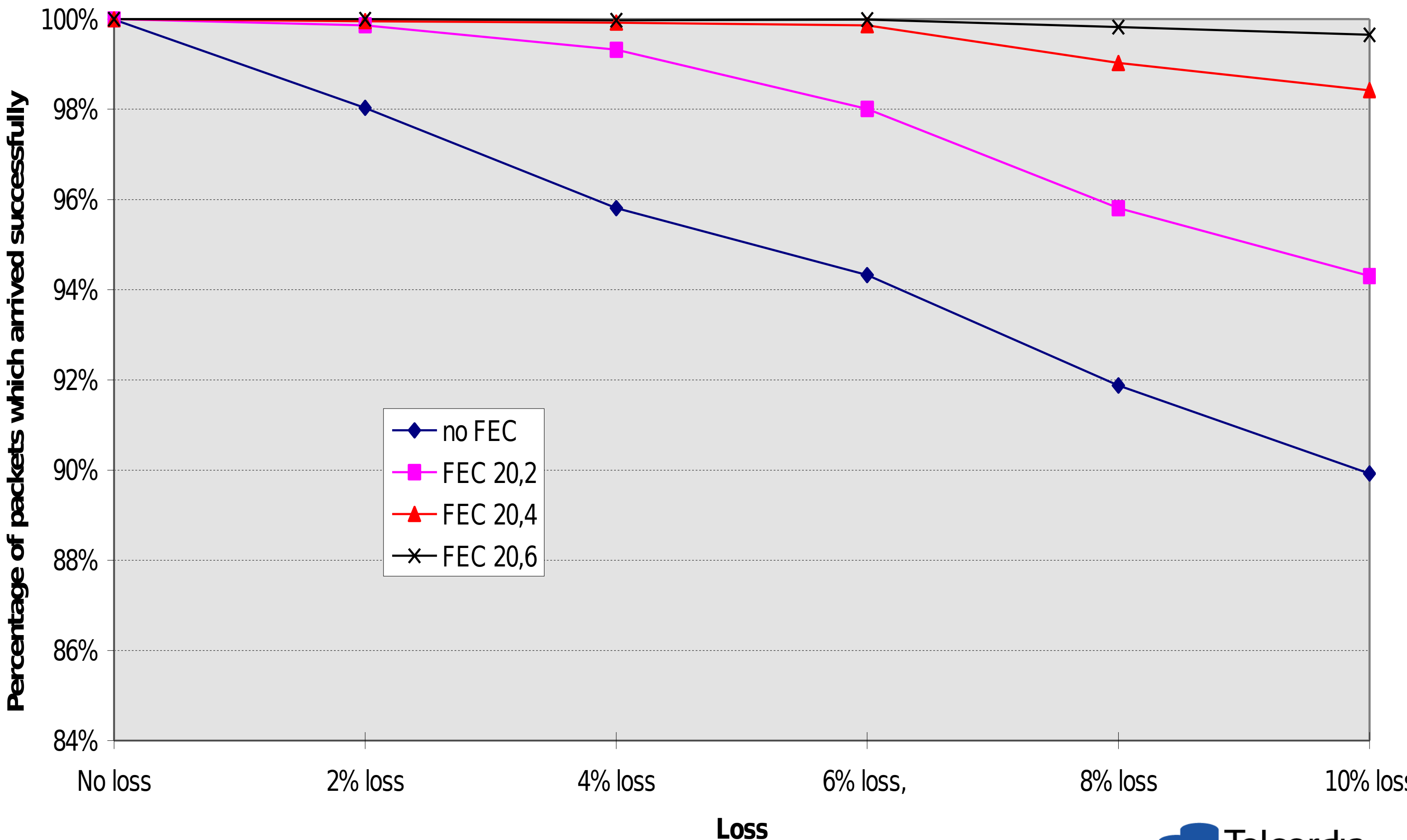
Protocol Boosters:
Experimental Setup



- Uses `ttcp` [with UDP and TCP options]
- F*C adds redundant packets
- Loss module removes packets
- FZC operates at access link line rates
- Unicast and multicast tested
- FEC operates at OC3c rates

Protocol Boosters:

Kernel-level Implementation: FZC with Random Errors

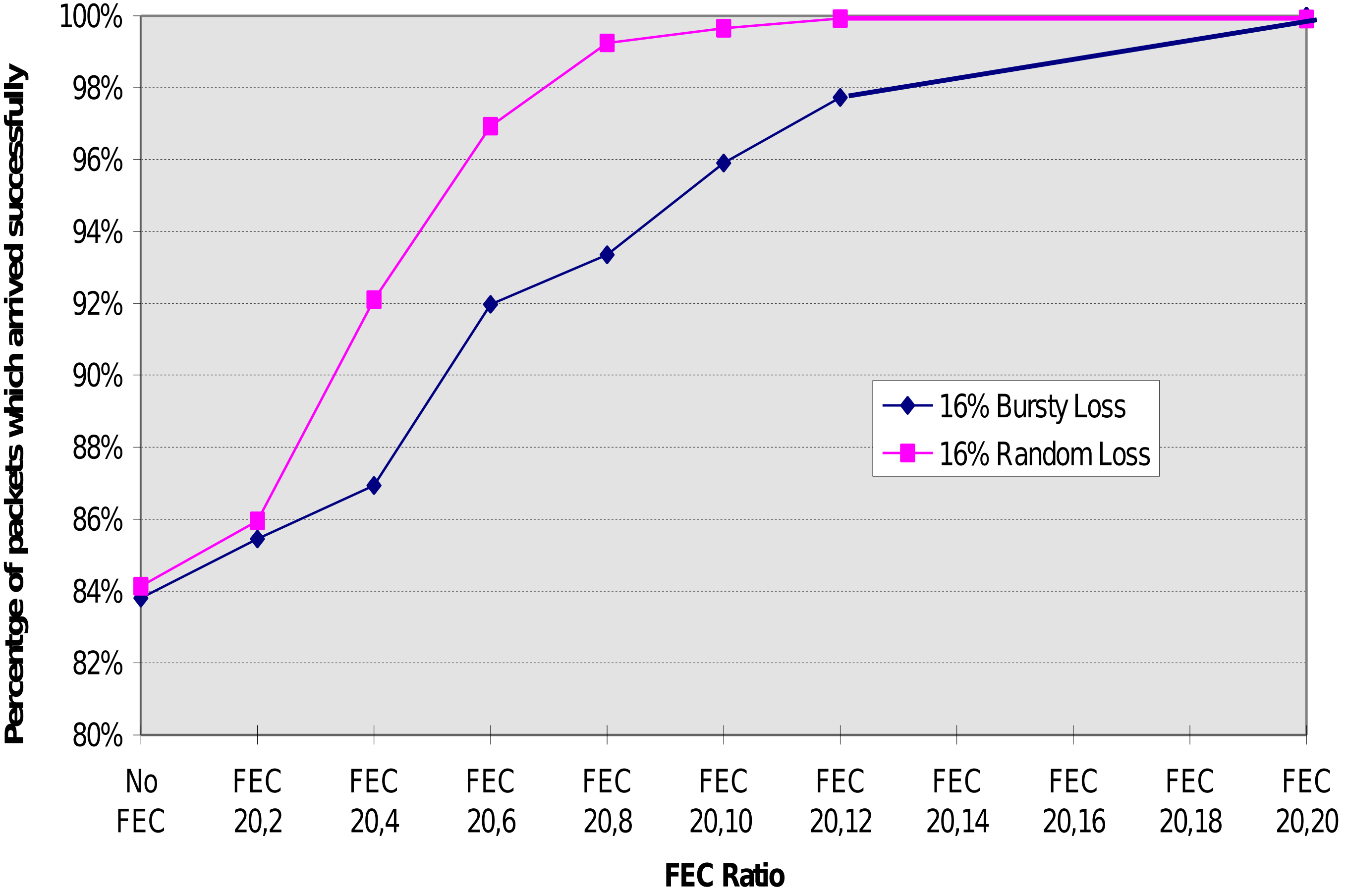


Telcordia™ Technologies

Formerly Bellcore…
Performance from Experience

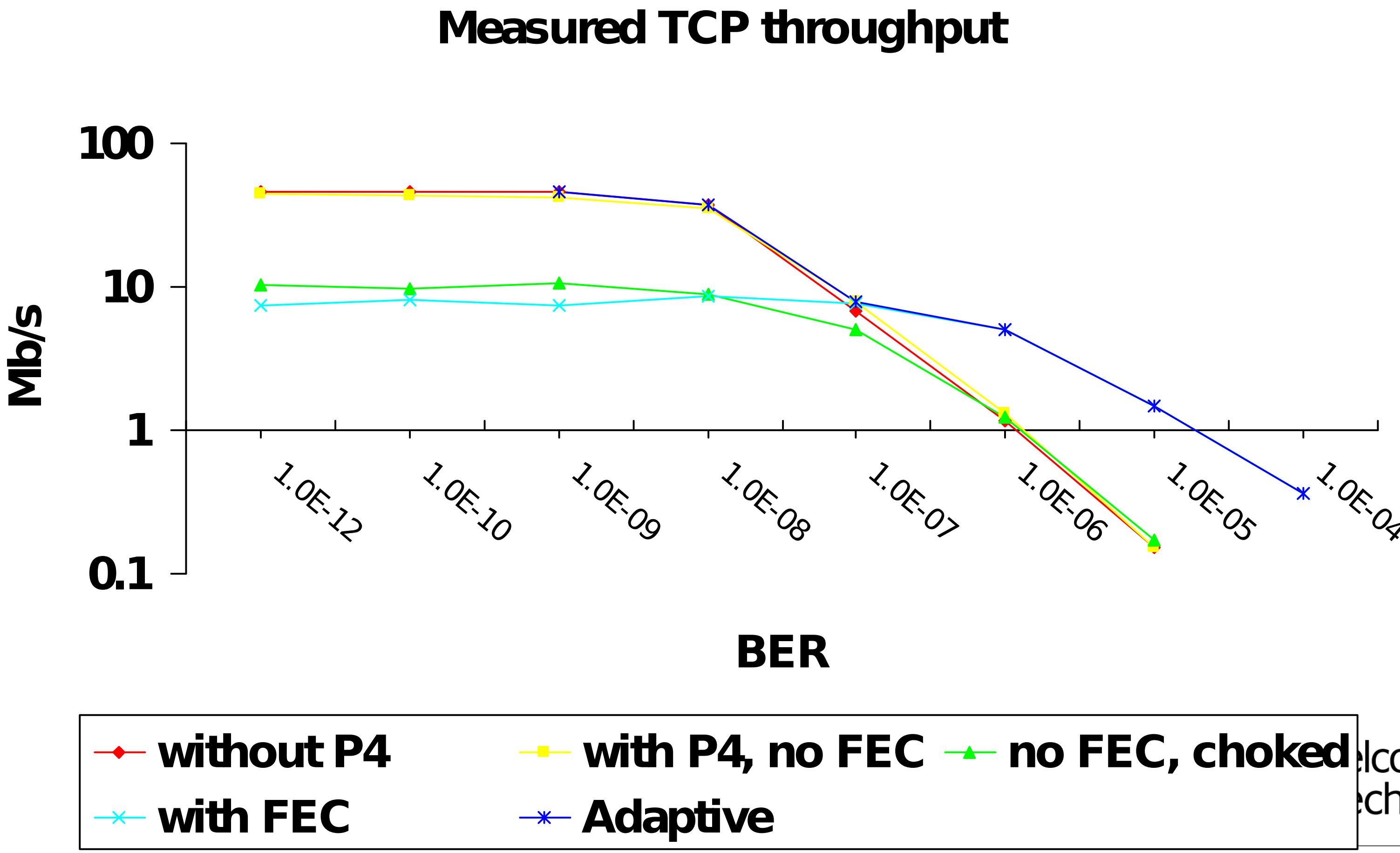
Protocol Boosters:

Kernel-level Implementation: FZC with Bursty Errors



Protocol Boosters:

P4 Implementation: FEC results

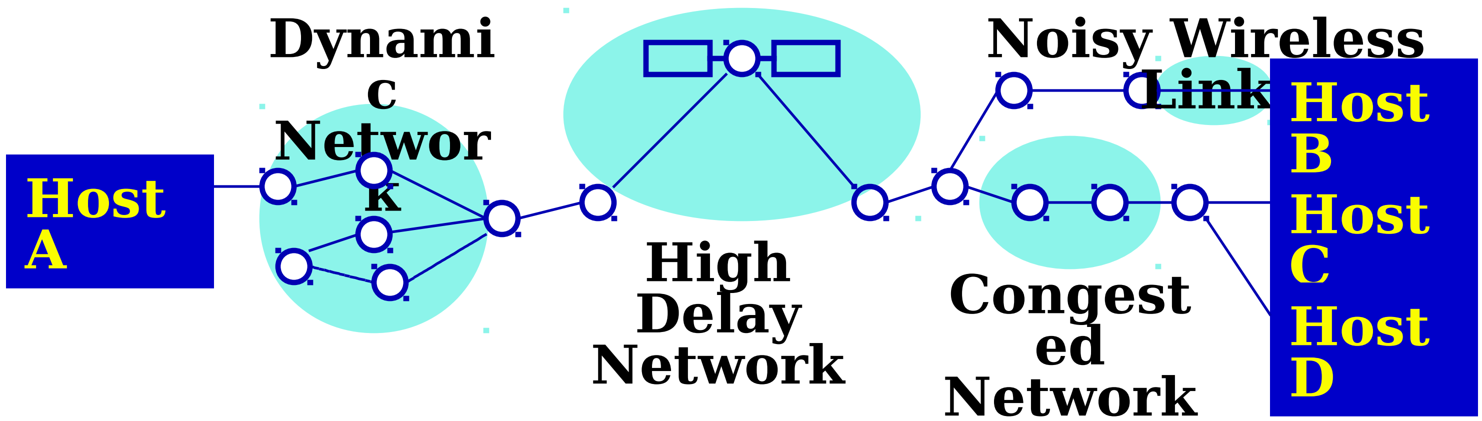


elcordia™
echnologies

Formerly Bellcore…
Performance from Experience

Protocol Boosters:

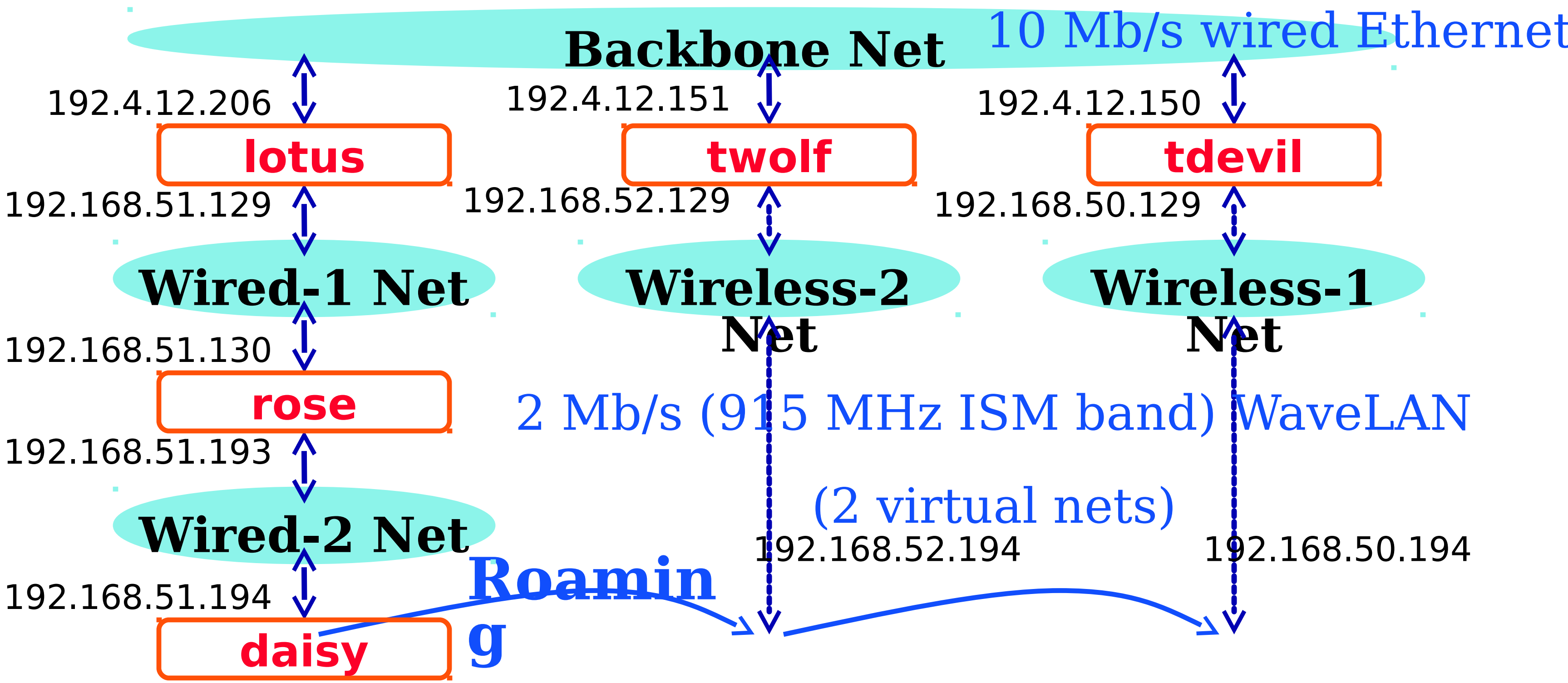
Kernel-Level: Demonstration QoS Testbed



Local retransmissions across noisy wireless access
Add (or retransmit) parity packets on multicast tre
Perform ACK manipulation over high delay networ
Local forwarding of packets to host in dynamic net
Applied selectively (e.g., mobile node signalling)

Protocol Boosters:

Kernel-Level: Demonstration QoS Testbed



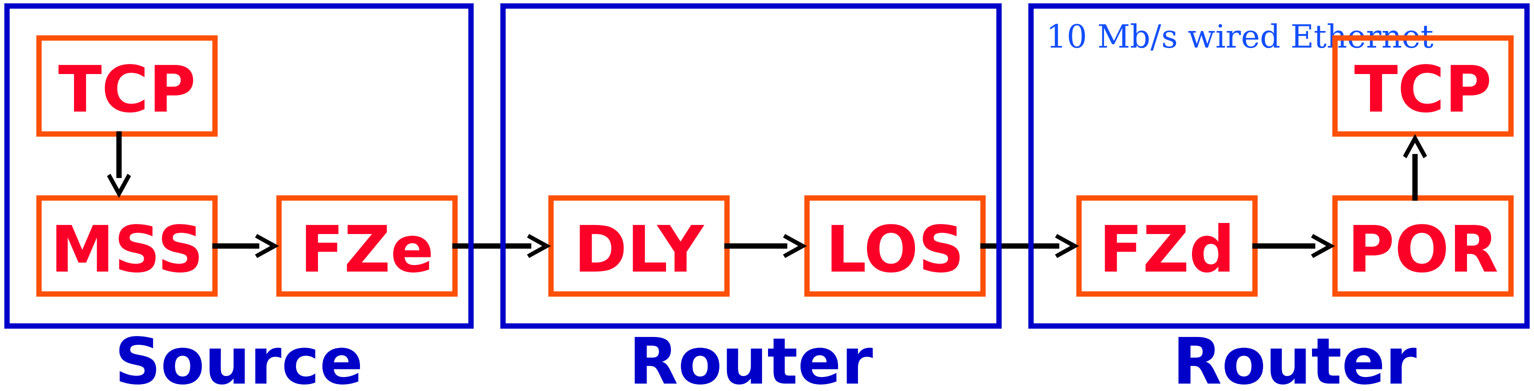
- 5 PCs running Linux 2.0.32 enhanced with:
  - **DVMRP** (xerox mrouted v3.9.beta3), **Mobile IP** (Stanford MosquitoNet v1.0.4), **DHCP** (ISC dhcpd v1.0)
  - **Protocol Boosters** (v0.3), **Multicast Proxy** (Bellcore v0.1)

Telcordia™ Technologies

Formerly Bellcore…
Performance from Experience

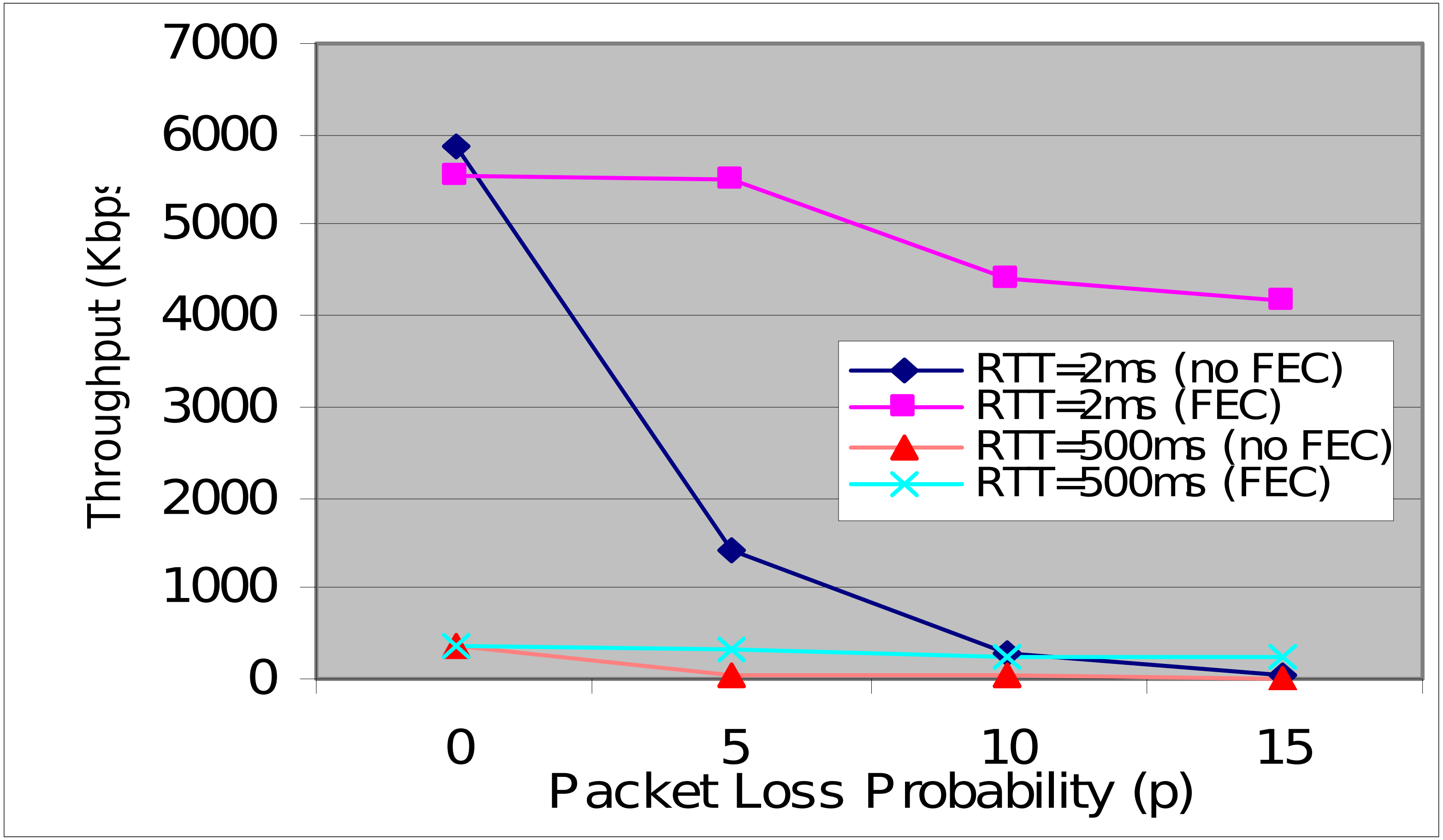
Protocol Boosters:

Kernel-Level: Demonstration QoS Testbed



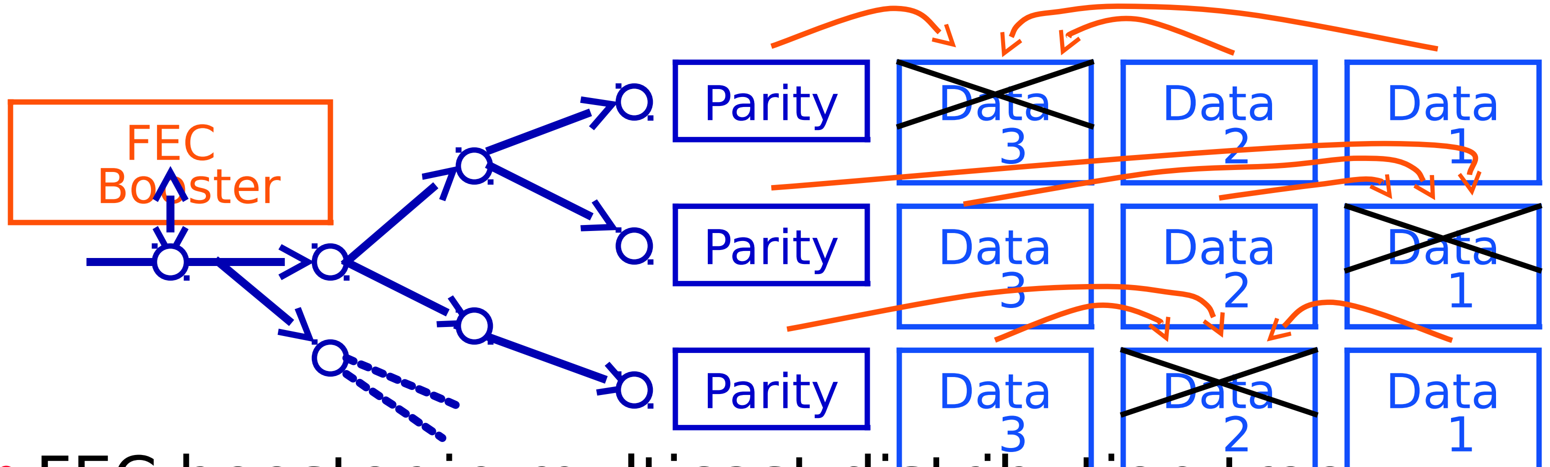
- 6 boosters used!
  - FZe adds parity packets; FZd regenerates data packets
  - DLY delays packets (sets RTT); LOS drops packets (sets p)
  - MSS reduces TCP’s Max Segment by 16 bytes (for parity)
  - POR limited reordering (to prevent triple-DUP ACK)
- TCP throughput is $O(RTT/p^x)$ where $x=1/2$ for small p

Protocol Boosters:
Kernel-Level: Demonstration QoS Testbed



Protocol Boosters:

Kernel-Level: Demonstration QoS Testbed



- FEC booster in multicast distribution tree
  - Transparently adds h parity packets (not change data packets)
  - Downstream receivers recover any h missing data packets
- Same parity recovers different packets at each receiver
  - Preemptively add parity packets (real-time applications)
  - Reactively retransmit parity (max number of missing packets)

Protocol Boosters:
Hardware-level: PMODs

General:

- Decide which booster is necessary
- Activate booster at right place and right time
- Coordinate dependencies among boosters
- Signaling

P4 Specific:

- Manage limited hardware resources
- Predict the need for booster and configure in advance

Protocol Boosters:
Hardware-level: PMODs

P4:

- BER-monitor based on AAL-5 CRC-32
- Checks AAL-5 packet
  - bad: X=1, good: X=0
- Calculates moving average:
  - $Y[n]=Y[n-1]-X[n-256]+X[n]$

Controller:

- Collects data from BER monitor (reads Y)
- Relates BER-monitor data with actual BER

Protocol Boosters:
Conclusions

- Devised an useful model for incremental protocol construction
- Demonstrated high-performance implementations of the model
- Demonstrated the value of Protocol Boosters for UDP and TCP applications
- Anticipate porting ‘best of the breed’ to SwitchWare/PLAN
- Technology Transfer to Army Research Lab Work (Airborne Communications Node - ACN)
- TCP performance improvements promising, but open issues
    - How adapt to different versions of TCP (e.g., when ACK)
    - Better understand FZC booster and TCP

Telcordia Technologies

Formerly Bellcore…
Performance from Experience